Modelling of Software intensive systems

Assignment 5 Devs

First Edition

L. Jason	Jason.Liu@student.uantwerpen.be	s0213082
S. Kin Ning	KinNing.Shum@student.uantwerpen.be	s0202054

2024 - 12 - 28

Contents

1 | Atomic Devs blocks

So here we will shortly explain every atomic dev we came up with. We have implemented 3 types of blocks:
1. Queue
2. LoadBalancer
3. Locks

1.1 | Interface

These blocks are connected to each other using different input and output ports. First, we have the inputs and outputs for the ships. These ports allow the ships to move through the system. Each block has 1 input and 1 output for the ships, except for the load balancer, which has a different output for each lock.

The queue doesn't immediately send the ships it receives to the load balancer. The balancer has to request a ship from the queue to receive one. This means that when the balancer needs a ship, it will send a request to the queue, with the size of the ship as its value. To know which sizes of ships are available, the queue outputs a list of integers, which represents the ship sizes, to the balancer. For effective load balancing, it is required to know which sizes of ships are available and only allowing ships through when there is enough capacity. This implementation allows that.

To know if the locks are open, the balancer internally keeps track of the capacity of each lock. It is also possible for the locks to close even when there is space left, that is why the lock will send its availability to the load balancer as a boolean. The balancer has an input port for each lock, allowing it to keep track of the locks individually.

In total, the queue has 2 input and 2 output ports, each lock has 1 input and 2 output ports, and the load balancer has 2 input, 1 output ports and 1 input and output per lock.

1.2 | Queue

The Queue atomic model manages queues of ships categorized by their sizes. It processes incoming ships, handles requests for ships, and notifies availability of ship sizes in the queue.

1.2.1 | External transition

When new ships arrive, we retrieve the ship from the input and check if this size already exists in our current queuing system. If not, we set the tag to true to notify later on that new sizes are available. This also happens when ships leave.

Next, we also have an input to take on requests. These requests are sent by the load balancer to retrieve a specific ship.

1.2.2 | Time advance

This function returns 0 if the load balancer has requested a ship, or if the available ship sizes have changed. Otherwise, it will return infinity.

1.2.3 | Output function

If the list of available ship sizes has changed, we output the new sizes that are available. If there is a request, we also have to send the requested ship to the load balancer. When the last ship of a size has left the queue, the new list of available sizes will also be sent, alongside the requested ship.

1.2.4 | Internal transition

After sending the new available ships, we clear this tag and also remove the requested ship from the queue.

1.3 | LoadBalancer

The LoadBalancer atomic model distributes ships to locks based on capacity and priority rules. It can prioritize larger or smaller ships depending on the configuration.

The load balancer is split in two types. We make use of one superclass and derive from this superclass to make two unique subclasses. This is because they both make use of the same four main functions, but they only differ in the type of algorithm they use.

1.3.1 | External transition

The external transition can take three types of inputs:

1. if ships are available
2. incoming ship
3. if lock is open or not

So in case the queue gets updated, we need to update the load balancer to let it know what is available.

In case we requested a ship before, we also have to take in this ship.

We also check if the locks are open. In case it is not, we evaluate that it has no capacity anymore. This is to say that no ship can enter anymore when closed (trivial).

After that, we utilize the load balancing strategy and make a request to the queue, by setting the time advance to 0. This also happens in the internal transition function.

1.3.2 | Time advance

We advance forward in case we need to process a ship, or we need to send a request. In case there is none, we just stay in this state forever.

1.3.3 | Output function

In case there is a ship and there is a lock, we can assign it too or there is a request we want to make.

1.3.4 | Internal transition

In case there was a ship, we have to change several things to our state. We have to change the next lock (need for round-robin) and the capacities that are available.

Just like in external transition, we apply the strategy to make a request to the queue.

1.3.5 | Algorithms

For both algorithm explained here, we use the same method to differentiate between priorities. In case we want smaller ships to have priority. We iterate from small to big first (vice versa for bigger ships).

1.3.5.1 | Round-Robin

The round-robin is a simple algorithm. We iterate through every lock and check if there is available space. If so, we set the needed variables for the request to the queue, and later we move this ship to the needed lock.

1.3.5.2 | FillErUp

The FillErUp algorithm is not that special either. We just check the difference for every ship and just check for the lowest difference.

1.4 | Lock

The Lock atomic model represents a canal lock that manages ships passing through. It waits until full or until a maximum wait duration is reached before allowing ships to pass.

1.4.1 | External transition

Here we can take in one input:
1. incoming ship

If a ship comes in, we update the capacity. In case there is no space left, we start the pass through duration. In case it is not full, we just wait for 60 seconds and start the passing phase.

1.4.2 | Time advance

The time advance is the amount of time that is remaining.

1.4.3 | Output function

When the lock is open, ships can enter and go out at this moment of time. So we output that the lock is open and all the ships that move to the sink.

In case the waiting time passes, we just close the lock.

1.4.4 | Internal transition

When passing is finished, we just assume that the lock is empty because we moved all ships to the sink. We set again the remaining time to infinite because there is no ship. We also set that we have full capacity.

2 | Results

After running these simulations, we have several results.

2.1 | Fillerup bigger

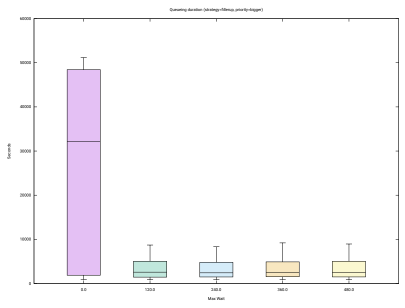

Figure 1: Box plot fillerup bigger

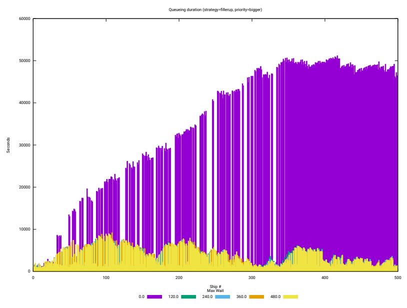

Figure 2: Ships fillerup bigger

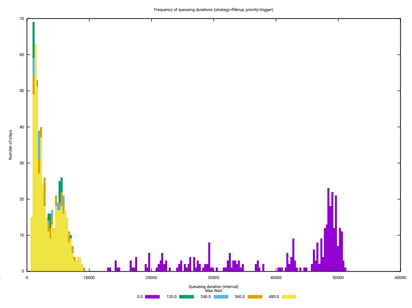

Figure 3: Frequency Graph fillerup bigger

When having a low wait time, the amount of ships that have to wait is drastically high. When adding intervals, this drastically decreases. No matter what waiting time we introduce, the results are all similar.

We can tell that having a higher waiting time, decreases the amount of ships waiting.

2.2 | Fillerup smaller

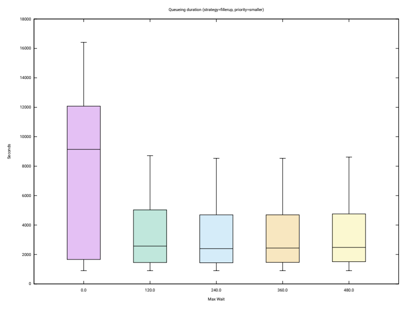

Figure 4: Box plot fillerup smaller

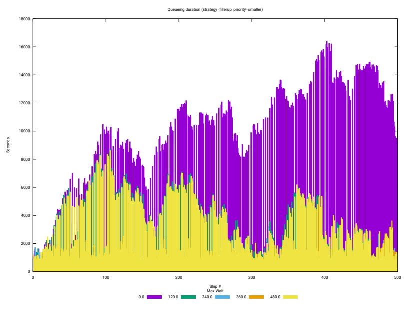

Figure 5: Ships fillerup smaller

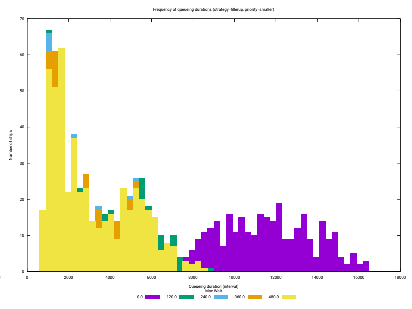

Figure 6: Frequency Graph fillerup smaller

When we prioritize smaller ships, the amount that needs to wait is higher. This is because there is higher chance that there is a gap. While prioritizing the bigger once first, has a higher chance of a smaller gap.

2.3 | Round-Robin bigger

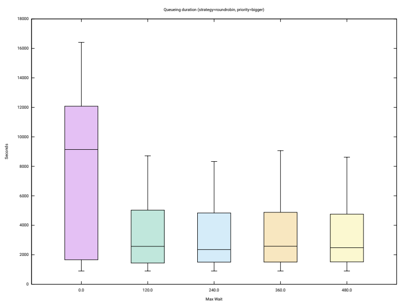

Figure 7: Box plot round-robin bigger

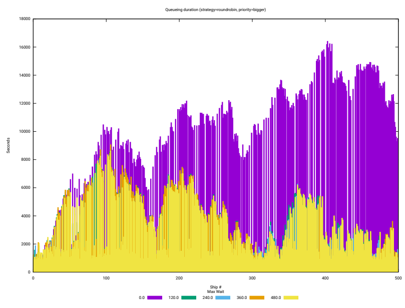

Figure 8: Ships round-robin bigger

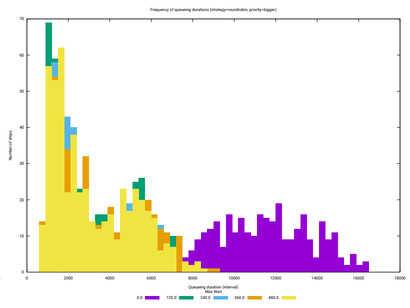

Figure 9: Frequency Graph round-robin bigger

We can see that applying round-robin with bigger ships results in similar results as fillerup smaller. This is probably because round-robin just takes the first next lock that fits, this can still causes gap that does not match it perfectly.

2.4 | Round-Robin smaller

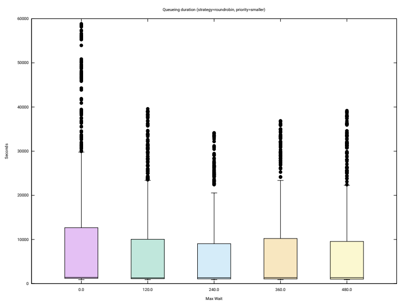

Figure 10: Box plot round-robin smaller

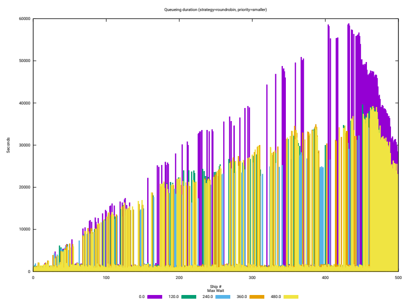

Figure 11: Ships round-robin smaller

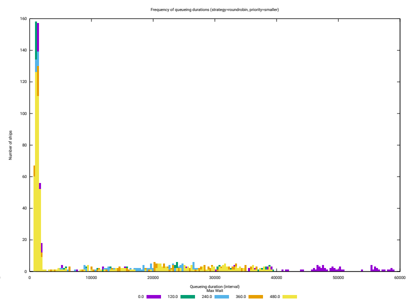

Figure 12: Frequency Graph round-robin smaller

Round-Robin with smaller ships is something special.

We can see in the box plot that we have many unique cases, we see many outliers. Even looking at the ships' waiting time, it is rather unique. Sometimes we have ships waiting a long time, and sometimes it is extremely low.

This is probably because sometimes it does fit perfectly, because there are mare ships that fit in to one lock. But there is probably also cases where there is a lot of space left but there is no ships that fit these spaces anymore.

2.5 | Conclusion

Depending on what situation we have we can get different result and depending on what we want we might want different algorithms.

We can tell that fillerup smaller and Round-Robin bigger behave the same, so choosing either of them would result in similar effects.

In case we have a specific amount of ships passing through the system, we can use Round-Robin smaller to balance our load.

In general, prioritizing bigger ships lead to less waiting time when ships arrive in intervals.